Going from Small to Large Data in Steganalysis

Ivans Lubenko and Andrew D. Ker

The Department of Computer Science, Parks Road, Oxford OX1 3QD, England.

ABSTRACT

With most image steganalysis traditionally based on supervised machine learning methods, the size of training data has remained static at up to 20000 training examples. This potentially leads to the classifier being under-trained for larger feature sets and it may be too narrowly focused on characteristics of a source of cover images, resulting in degradation in performance when the testing source is mismatched or heterogeneous. However it is not difficult to obtain larger training sets for steganalysis through simply taking more photos or downloading additional images.

Here, we investigate possibilities for creating steganalysis classifiers trained on large data sets using large feature vectors. With up to 1.6 million examples, naturally simpler classification engines must be used and we examine the hypothesis that simpler classifiers avoid overtraining and so perform better on heterogeneous data. We highlight the possibilities of online learners, showing that, when given sufficient training data, they can match or exceed the performance of complex classifiers such as Support Vector Machines. This applies to both their accuracy and training time. We include some experiments, not previously reported in the literature, which provide benchmarks of some known feature sets and classifier combinations.

Keywords: Steganalysis, Benchmarking, Online Learning, Large Data Sets, Average Perceptron, Support Vector Machine

1. INTRODUCTION

Most image steganalysis is based on supervised machine learning methods, and the paradigm has been largely unchanged for the past decade: a feature set of a few dozen to a few hundred features extracted from every image, and a nonlinear classification algorithm (almost always a kernelised Support Vector Machine) trained on a few thousand examples.[1–7] Although there have been some exceptions to this, including alternative classifiers,[8,9] the use of regression instead of classification,[10] and notably the introduction of very large feature sets and ensemble classifiers,[11] the size of the training data has remained fairly static, usually between 4000 and 20000 training examples (half cover and half stego). For the larger feature sets this means that the classifier is under-trained, and so may be too narrowly focused on characteristics of a particular source of cover images, with performance degrading when the testing source is mismatched or if the source is a heterogeneous mixture.[2,12,13]

Unlike the application of machine learning to, for example, natural language processing or intrusion detection, it would not be particularly difficult to acquire larger training sets for steganalysis: one simply takes photographs (or downloads more images) to get more cover data, and embeds data at will to create stego data.

In this paper we investigate some possibilities for creating steganalysis classifiers trained on large data sets (up to 1.6 million examples), with large feature vectors (around 50000 features per image). The sheer size of the training data means that simpler classification engines must be used, and this also allows us to examine the hypothesis that simpler classifiers will avoid overtraining, and thus perform better on realistic heterogeneous data. We highlight the possibilities for online learners, which can be incrementally trained. We perform some experiments to demonstrate that extremely simple online learners, when given sufficient training data, can match or exceed the performance of complex classifiers such as Support Vector Machines (in terms of both accuracy and training time). This includes some experiments which provide benchmarks of some known feature and engine combinations which have not been reported in previous literature.

Further author information: (Send correspondence to ADK):
A. D. Ker: E-mail: adk@cs.ox.ac.uk, Telephone: +44 1865 283530
I. Lubenko: E-mail: ivans.lubenko@cs.ox.ac.uk

2. MOTIVATION FOR THE “LARGE DATA” APPROACH

In recent research, steganalysis classifiers tend to work with rather small training sets, compared with the number of features. A typical testing scenario includes one or more image sets from a unique cover source (personal photo library or a public image database such as the NRCS photo gallery) and the results are reported on per-set basis. The size of such individual homogenous data sets varies between 2000 (Ref. 5) and 10000 (Ref. 14) images. The actual number of training examples is then doubled because we normally consider the binary classification problem where cover images are to be discriminated against the images carrying the stego signal of a fixed payload size. Unlike the image sets, the size of the feature sets has been steadily increasing. For example, a near-50000-long feature set was recently introduced for detection of nsF5.[11] For such problems many accurate classifiers have been found,[8, 11, 15] notably the kernel Support Vector Machine (kSVM).

The performance of these typical classifiers is often disappointing on heterogeneous data sets, or if “cross trained” (trained on data from one source, tested on another). It has been shown that combining the data sets negatively impacts the success rate of detection.[2, 12, 13] Pevny shows that training a model on images from four sets lead to the classification error increasing two-fold (ref. 13, Tables II and V). A similar result was achieved in Ref. 12. In the real word, however, a steganalyst is likely to be required to deal with images from a mixture of sources. Therefore it is desirable to find good classifiers for heterogenous training and test data, rather than training and testing our classifiers on simple data from one source. Fridrich[16] suggested that the problems of poor generalisation on non-homogenous data occur because the steganalysis features like those used in Refs. 2, 12, 13 (known as WAM, CC-PEV, and SPAM, respectively) are sensitive to microscopic properties of the source. It is unclear whether the problems persist when using larger feature sets (e.g. CC-C300, MINMAX+MARKOV[11]) or larger data sets (orders of magnitude more than 20000), or whether other factors exist that negatively impact the classification success rate in the heterogenous data test case.

However, it is not so difficult to obtain large training sets for steganalysis. We can just take more photos, and other operations such as data embedding, feature extraction and labelling require no manual processing and are cheap, compared to training a classifier. Here, we test on data magnitudes larger than that have been used in previous research, e.g. Ref. 1–7, 14. Our hypothesis is that such a *large training set will allow for training a less cover source dependent classifier*. Classifiers trained on large non-homogenous data should, in our view, generalise better on the test examples than complex classifiers trained and tuned to small data sets. This is similar to the ideas exploited by Yahoo! Labs which they advocate in Ref. 17. It has been found that using large data improves the performance of machine learning algorithms. In Ref. 18, the author gives examples of machine learning research where large data made a positive impact on the empirical results from experiments. In particular, data sets containing over a million instances were employed. We take our cue from these examples and propose to train on data of the same magnitude.

A further reason for the use of bigger training data sets is the recent increase in the size of steganalysis feature vectors. The state-of-the art feature sets for both transform (CC-C300) and spatial (MINMAX+MARKOV) domain are very large[11, 14] and non-sparse, so they require larger data sets to populate the feature space diversely. Also, research has shown[19] that large dimensionality imposes a higher chance of overfitting.

Unfortunately, complex classifiers such as the kSVM are limited in how much data they can train on, because of the computational complexity involved. In the next section we consider the difficulties associated with working with large data.

3. WORKING WITH LARGE DATA SETS

We are looking into using millions of examples for our training data set. In order to acquire such data for steganalysis we need a source of public images. They are not required to share some common properties such as those that affect within-image statistics because we are concerned with the real-world problem, i.e. a non-homogenous image set. But certain properties, such as format and size, should be fixed for simplified processing and embedding. The Internet contains many such images and we will use images downloaded from public sources in our experiments.

However there exist certain difficulties when dealing with data of this size. If steganalysis were performed using, say, 1000 (double-precision floating point) features per image, with a training set of 20000 images, the raw training data would take a mere 160MB to store. Large data presents new difficulties: in our experiments we will use around 50000 features per image, and in the extreme case up to 1.6 million training images, which means 640GB of training data. This is far too large to fit into the memory of a modern computer, and at typical SATA 2 transfer rates of 100MB/s would take ten hours just to read from a hard disk. Any algorithm which is to process such data must be extremely simple, preferably passing just once through the training data.

3.1 Simple Supervised Classifiers

Complex algorithms such as the kernel SVM are too slow for large data. Most optimisation algorithms associated with such complex classifiers, e.g. the (quasi-) Newton's method (SVM) or the gradient-descent (Logistic Regression[8]), currently offer only super-$O(N)$ time complexity. In particular, kSVM's time complexity is $O(MN^2)$, where M is the number of features and N is the size of the training data. Taking into consideration the requirement of parameter optimisation, the practical computation time of such algorithms increases even further. This limits the size of data that can be processed, which has been shown in Ref. 11, where kSVM was reported to scale poorly with the number of features given a fixed data size.

Linear time complexity, which could be gained from using a simple algorithm, would be very helpful for training on large data. The Perceptron is an example of such a simple linear algorithm that is very fast. Its learning rule is very simple involving only a few operations to update the weight vector. The Average Perceptron is a regularised version of it that improves its stability and performance. We will use the Average Perceptron to process our data set of 1.6 million training examples. Visiting more data points allows for training a more general model, which has the potential to perform well on test data.

3.2 Online Algorithms

Online learners are those which continually update their model, as more training data is supplied, without having to remember past training data. They process examples one by one in an incremental manner using stochastic update. The stochastic update rule approximates the online classifier's objective function by estimating the loss incurred after seeing only one example. The ability to learn piece-by-piece may itself be an advantage, if the steganalysis application requires a continually-refined cover model.[20]

One feature which is critical for classifiers trained on very large data is to pass through these data only once, and online algorithms provide this. It was shown that $O(1)$ passes over a dataset are sufficient to obtain optimal approximation.[17] There exist online versions of popular optimisation algorithms such as the gradient descent, which is a part of common classifiers such as Logistic Regression.[8] However we will look at linear online algorithms, in view of their simplicity and speed. Unlimited data can be processed because the trained model can be saved at each incremental step. Moreover, "[we can reduce] the runtime behaviour of inference algorithms from cubic or quadratic to linear in the sample size".[21] There are additional benefits as well, importantly the absence of parameter optimisation and the ability to test the classifier as we train.

4. EXPERIMENTS

4.1 Acquisition of Large Dataset

Our experiments were performed on a highly realistic data set, obtained from a leading social networking site. This site allowed users to make their photos open to the entire public, if they wished, and gave viewers the option to click "next photo" links to see the entire set of photos featuring each such user. Also, it offered links to photos of other users, tagged in the current photo, who chose to make theirs public too. Many users made their photographs public. By starting with a single public photo, and automating the process of clicking on these links, we downloaded over 4 million publicly-visible JPEGs from the site during December 2010. The crawl was restricted to users who publicly identified themselves with the Oxford University network and the data was then anonymised. We emphasise that only publicly-available photos were downloaded, and that no personally-identifiable information was retained.

Each photo is identified with the unique user who uploaded it. This enabled us to select a subset of exactly 200 photos from each of 4000 users. These 800000 images form the data set used in this paper. It is highly realistic because it is exactly the sort of media used on the internet in large social networks. Our only caveat is the small possibility that some of the users were already using steganography in their images: if so, our set of "cover" images may not be entirely clean, and may contain a few stego images. Hopefully, use of steganography in social media is not (yet) widespread, at least in Oxford University.

These images are very difficult for conventional steganalysis: the social network offers to resize uploaded images to approximately 1 megapixel (but by no means to a uniform size), and also recompresses almost every image to JPEG quality factor 85. This means that the cover images will have resampling and double-compression artefacts, the latter known particularly to affect steganalysis reliability.[15] The different users will be using different cameras, which makes the set heterogeneous (except for its ultimate compression factor, and restricted range of images sizes). Some images are not even natural photographs: they may be montages, have captions, or be entirely synthetic. In a realistic scenario, we must deal with this type of difficult data.

The only way in which the data set has been screened was the removal of images with file size smaller than 5KB (which deletes images with little or no content) or which do not follow the standard compression factor (this was less than 1% of the images downloaded).

4.2 Embedding

We focus on the nsF5 embedding scheme[22] with payloads of 0.1 and 0.2 bits per non-zero Discrete Cosine Transform (DCT) coefficient (bpnc). The nsF5 algorithm is the improved version of F5 that uses wet paper codes to eliminate shrinkage.[22] nsF5 is a simple algorithm, probably one of the best for JPEGs and we do not have any reason to think that the choice of embedding will make a significant difference to our conclusions in this work. We do not have sufficient disk space to extract features for too many different algorithms, but this will be the subject of future work.

Training and test sets contain 50/50 split between cover and stego images, with stratification of cover-stego pairs.

4.3 Extraction of Features

We employ the state-of-the-art 48600-dimensional CC-C300 feature set which was a part of the slightly larger feature set that currently gives the most sensitive detector of nsF5 in JPEG images when coupled with an ensemble Fisher Linear Discriminant.[11] A brief comparison was drawn with the 548-dimensional CC-PEV features in Ref. 11, where only the ensemble classifier's figures were reported for both feature sets. We aim to report the kernel SVM figures as well because there is little research into comparison of the CC-C300 set to other transfer-domain features within the framework of one classifier.

4.4 Training Methods

Our aim is to test our hypothesis by comparing the accuracies of a number of classifiers on non-homogenous test data, where possible trained on the entire data set. We will test online and iterated versions of Average Perceptron, ensemble methods based on the Average Perceptron and the Fisher Linear Discriminant (FLD) and compare them to kSVM as the state-of-the-art reference point. A review of each of these algorithms is given below.

4.4.1 Support Vector Machines

Kernel SVM was designed to produce high accuracy from small data through the use of the kernel trick and the soft margin. These make it very suitable for a typical steganalysis problem, as we know it, i.e. a binary classification problem with very limited training data. But they are not ideal for large training data.

We use the RBF (Gaussian) kernel, following the configuration of the original SPAM detector[13] and Ref. 8. In this configuration, the kernel SVM has two parameters to optimise, cost C and kernel width γ. A grid search is performed to find the best combination of C with γ, with five-fold cross validation also used to minimise statistical noise. The grid in our experiments was experimentally reduced to the following 6 values of C from

$$\left\{2^i \mid i \in \{11, 13 \ldots, 21\}\right\}$$

(with closer examination around the best values), and 6 values of γ from

$$\left\{2^j \mid j \in \{-13, -15, \ldots, -23\}\right\}.$$

Whilst very simple to use because of the availability of ready tools such as libsvm,[23] kSVM is impractical for real-world problems such as the near-online problem described in our experiments. Using the large modern feature vector, CC-C300, it was only possible to process a maximum training set size of 20000 and only using a very limited grid search. Anything larger simply takes too long for any useful purposes. It has other disadvantages too: unlike online methods it is iterative and hence requires many passes through the data and also requires parameter tuning via the expensive grid search.

We used the libsvm[23] with its extensive toolkit for our kSVM tests.

4.4.2 Ensemble FLD

For the ensemble algorithm we will follow the example in Ref. 11. In this configuration, L Fisher Linear Discriminant base learners were trained on L different subsets of k dimensions drawn at random from the original CC-C300 feature set. Here, as in Ref. 11, $L = 99$ and $k = 2000$. The base learners' outputs were then combined for classification using the majority vote:

$$y(x) = \left\{ t \mid t \in \left\{-1; +1\right\} \wedge \underset{t}{\operatorname{argmax}} \sum_{l=1}^{L} \left(w_l^T (x - c_l) = t\right) \right\}$$

w is the Fisher's discriminant, c is the parameter that guides the position of the hyperplane and can be found analytically and t is the label of training example x.

This method has shown[11] very promising results using the new CC-C300 feature set and its derivative feature sets. A major advantage was shown to be the speed of training and the reduced complexity of parameter optimisation. No direct comparison was drawn with kSVM using the same features.

4.4.3 Average perceptron

We will use an online version of the Average Perceptron,[24] which is a self-regularising version of the Perceptron algorithm. The Averaged Perceptron is capable of training on one example at a time, which allows for processing unlimited data with no memory overhead.

This is made possible through the simple update rule it shares with the Perceptron, which only requires the weight vector and the feature vector of the current image. The Perceptron aims to minimise the number of incorrectly classified examples, i.e. the training error:

$$\min(E_p) = \underset{w}{\operatorname{argmin}} - \sum_{n \in M} w^T x_n t_n$$

where w is the weight vector, $t \in \{-1; +1\}$ is the label of training example x and where M is the set of all misclassified training examples. The minimisation is realised via stochastic updates, which allow for approximating w one example i at a time:

$$w_i = w_{i-1} + x_i t_i$$

The update happens when a new input example is assigned the wrong label. In the Average Perceptron the update includes the regularisation step, where the average weight vector is updated:

$$w_{avg} = w_{avg} + w_i$$

The vector w_{avg} is used in the final decision function to predict the label of test example x:

$$y(x) = sign(w_{avg}^T x)$$

4.4.4 Batch iterated perceptron

Batch iterated perceptron is our modification to the Average Perceptron algorithm, where data is processed in batches of fixed size n. In this algorithm we save n consecutive input examples and instead of processing the next example $n+1$, we iterate over these n examples k times, for some fixed k. This procedure is then repeated with the next n input examples, this time starting from example $n+1$ and so on. The averaged weight vector w_{avg} is still updated stochastically throughout the training. The idea behind this algorithm is to train the online model on same examples multiple times to maximise the chance of convergence, which, in our experiments, the Averaged Perceptron seems unable to achieve even after 1.6M training examples.

In our experiments we used the batch sizes of $n = 4000$ and $n = 40000$, and a fixed number of iterations $k = 50$.

4.4.5 Ensemble average perceptron

We also introduce here the third type of large-scale classifier: the online ensemble Average Perceptron, which fuses the advantages of stochastic nature of online learners with the increased accuracy of ensembles classifiers. The set up remains identical to the ensemble FLD, with the same number of learners, same number of randomly-chosen features per learner, and majority vote, except that stochastic learning is made possible via the use of online base learners each employing the online Average Perceptron update.

4.4.6 Implementation considerations

Due to the sheer size of our data sets, they cannot be processed all in memory, which poses problems for many learning algorithms, even when working in the reduced subfeature space. In particular we found that:

a) Iterative algorithms are slowed down by disk access because the processing is only possible in chunks; this requires careful memory management. In particular, on our machines the iterative Average Perceptron was trained on chunks of up to 50000, but this figure may be required to be reduced if memory is more limited (our main experiments were performed on a machine with 96GB of memory).

b) Kernel SVM can be very space and time demanding, specifically on the more complex problems. The kernel trick requires extra memory space and it is difficult to estimate this requirement. Problems of different complexity (0.1 bpnc or 0.2 bpnc) as well as certain choices of the γ and c parameters affect both kSVM's space and time complexity. For example we found that training it on 0.1 bpnc took five times longer than on 0.2 bpnc. Necessarily this limited our training set sizes for this classifier. In practice it was found that kSVM required data set sizes greater than 10000 examples to compete with other algorithms, which quickly became too large to use in the time available for our experiments, given the requirement for the grid search and the extra large feature set. Similarly, even doing just one training run on much larger data was found to be impractical as well.

c) Ensemble FLD also calls for storing the full training data in memory unless the reduced feature sets are pre-calculated for the whole ensemble. Even though we were able to test ensemble FLD on 100000 and 400000 examples, it was only possible by precalculating and storing the subfeatures on disk. This preprocessing procedure drastically reduces the computational efficiency of the algorithm.

These problems show that online learning is desirable when processing large and high-dimensional data.

4.5 Benchmark

The classifiers will be evaluated on testing sets which are completely separate from the training data. There are two such sets, one for each payload size. Both sets are composed of a set of 40000 images with 50/50 stratified cover/stego split and drawn at random from the same source as the original training data.

Normalisation of the data was performed to zero mean and unit variance: this is critical for the success of SVM[23] and perhaps other classifiers as well. It is less clear whether normalised data affects the ensemble perceptron or ensemble FLD algorithms, but from our experiments we found that they perform better on normalised data when tested under the same conditions. The normalisation was based on the testing set only, as it would be very difficult to perform accurate normalisation on the training data in the online setting (i.e. approximating the normalisation factors as training examples arrive). Two different normalisation coefficients were required, one for each payload.

Our testing benchmark is the testing accuracy, which is defined as the proportion of correctly classified examples measured on the testing set. We also measure the approximate training times, which are reported in the results section.

The majority of our experiments were performed on the main test machine: a 12 core Intel Xeon 3.47GHz with 96GB of RAM. Some of the memory-intensive experiments with kSVM and eFLD would not have been possible without it.

5. RESULTS

Figure 1 (top) shows the summary of the experiments we performed, using all five classifiers, on the 0.1 bpnc testing set. Note the nonlinear x-axis. The lines represent the test accuracy of the online training algorithms, as they pass through the entire 1.6M training examples, whilst the individual marks show the accuracy achieved by the other algorithms given the indicated training set size. Each of the non-online algorithms had to be trained on the first K examples, where indicative values of K were chosen subject to feasibility of performing the training in a reasonable amount of time. The horizontal threshold line shows the accuracy that was achieved by the ensemble version of the Average Perceptron once it had used the entire 1.6M training set.

The corresponding results for 0.2 bpnc embedding are shown in fig. 1 (bottom): the results are in line with those for 0.1 bpnc and will not be discussed further.

Almost all experiments use the CC-C300 feature set, but we also performed a few using the CC-PEV features[4] for comparison with prior art. Note the relatively poor performance of CC-PEV features on this data (0.892 accuracy on 0.1 bpnc payload, with 10k training examples). This illustrates how difficult the social networking dataset is, with heterogeneous images and double-compression artefacts. It seems to be an emerging phenomenon that simpler, less specialised, feature sets perform better on heterogeneous data.

The red lines shows clearly the nature of the linear Average Perceptron algorithm. Despite its attempt at self-regularisation, its instability yields the rough line which does not appear to converge on this data. However it still manages to yield accuracy very near the benchmark line, given enough examples. This highlights the presence of linearity in this classification problem when expressed using the full CC-C300 feature set, and perhaps enforces the motivation for using fast linear classifiers.

We expect that a different choice of a quick linear classifier here, e.g. Logistic Regression with online gradient descent optimisation, would have resulted in a more stable performance, but one that was similar in terms of the overall accuracy performance. We suspect that the spikes of instability owe to some outliers in the training set, such as computer-generated images or images containing computer-generated content. A task for the future will be to implement *active learning* with Average Perceptron, where such outliers would be ignored during training. The black line was produced by our new online ensemble algorithm using the Average Perceptron. It shows an ability to converge relatively quickly and is the best performing of any of the classifiers we tested, though it does need on the order of 100000 items of training data to achieve this good performance. The high accuracy demonstrated here in conjunction with the fast unlimited training and the absence of tuning makes it a good candidate classifier for steganalysis if large training data is available.

Although the kSVM appears to make the most of a small training data set, its advantage is no longer evident when compared with an online nearly-linear classifier such as the ensemble Average Perceptron that has been trained on a larger data set. We were only able to train a kSVM on a maximum of 20000 examples when using the full CC-C300 feature set, an experiment which took over 10 days. With kSVM (shown as crosses on the graph) the accuracy was achieved by using hyperparameters from a grid search. Inevitably time and skill needs to be invested in finding the suitable parameters, which is both costly and prone to human error. With slightly different parameters the kSVM accuracy could have been significantly lower or significantly higher depending on which points on the optimisation plane are visited. However without visiting infeasible number of points we would not be able to guarantee kSVM hitting its optimal accuracy. And, as it is demonstrated in the diagram, this happens only with the largest training data set, which is the slowest to optimise for.

Shown in circles on the graph was the Average Perceptron run in iterative mode. When trained on chunks of data in 50 iterations it exhibited a quicker tendency towards convergence than the online Average Perceptron, but was a somewhat behind other classifiers when compared in terms of their ultimate accuracy. The reduced accuracy could perhaps be explained by undertraining due to the small iterations count, something which requires further investigation.

The comparison of runtime behaviour of these algorithms also shows the advantage of simple online classifiers. With our implementation, it takes approximately 10 days to train a kSVM on 20000 samples with the CC-C300 features, even using a reduced grid search for the hyperparameters. On the other hand, it takes 1 hour to train an Average Perception on 1.6M samples. Our implementation of the Kodovsky's ensemble FLD takes about 8 hours to train on 20000 samples, and over 7 days on 400000 (which can be potentially reduced slightly using hyper threading). It takes about 7 hours to train the online ensemble Average Perceptron on the full 1.6M data set. In general it appears to be faster to train a simple algorithm on a very large data set, than a complex algorithm on a small data set.

In summary, it can be safely concluded that the "generic" CC-C300 features have performed better than the specialised CC-PEV features for classification of non-homogenous data embedded with nsF5. For kSVM to return good results, it appears to require training on data set too large to process in practice. The online algorithms are capable of achieving the same performance in much shorter time but (so far) only in the ensemble setting. Potentially the "iterative batch" set up for online Average Perceptron would also be capable of achieving this benchmark accuracy even faster, however this needs further investigation using more iterations. In terms of the size of training set, the ensemble Average Perceptron requires a few hundred thousand examples to converge, whilst the batch iterated Average Perceptron will potentially require more training data, based on the findings from our current experiments.

6. CONCLUSION

In this paper we have investigated a new type of classifier for steganalysis: the online algorithms, in particular some varieties of the Average Perceptron. We have benchmarked their performance, using the large CC-C300 feature set, against other state-of-the-art classifiers, on a realistic heterogeneous data set of images from a social networking site. These images present difficulties for classical steganalysis because of their diverse character.

We found that the simple linear algorithms that we employed can be equally as accurate as the complex ones. These simple algorithms require large data to achieve such performance and must be online. However they are significantly faster to train on such large data than the complex algorithms are on smaller data sets. It was also noted that using simpler but larger "generic" feature sets combined with the simple and, in this case, linear algorithms can do better than targeted ones. We suspect that this is because the "generic" features remove the non-linearity from the steganalysis problem. Simple algorithms are required to work with these features effectively and avoid overtraining.

Whilst for some scenarios obtaining large training data is unrealistic, for many scenarios it is a distinct possibility. We suggest that where large training data has been obtained for real-world like data sets tackling the them with online algorithms is most suitable.

Of course, for robust conclusions we should explore the parameter space of the ensemble classifiers and further examples of online and perhaps active learners. This is the main direction for further research.

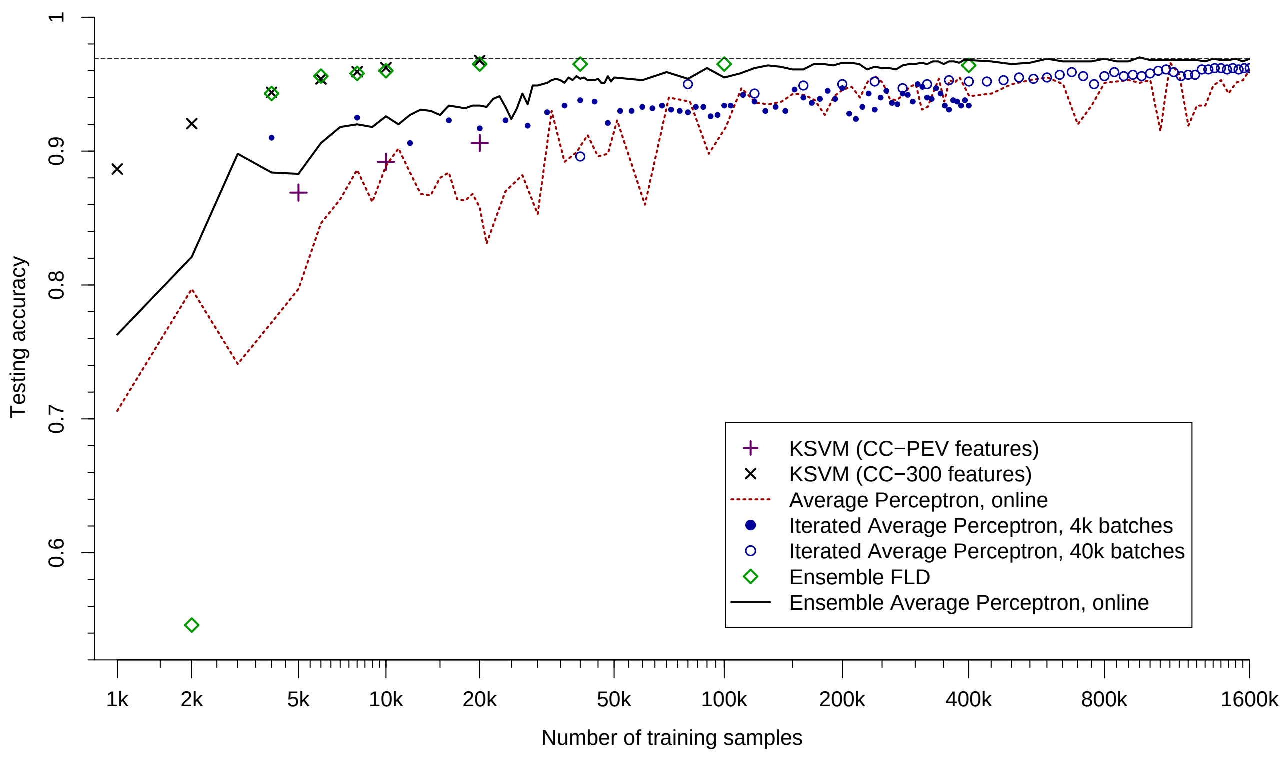
REFERENCES

[1] Farid, H. and Lyu, S., "Detecting hidden messages using higher-order statistics and support vector machines," in [*Proc. 5th Information Hiding Workshop*], *Springer LNCS* **2578**, 340–354 (2002).

[2] Goljan, M., Fridrich, J., and Holotyak, T., "New blind steganalysis and its implications," in [*Security, Steganography and Watermarking of Multimedia Contents VIII*], *Proc. SPIE* **6072**, 0101–0113 (2006).

[3] Shi, Y. Q., Chen, C., and Chen, W., "A Markov process based approach to effective attacking JPEG steganography," in [*Proc. 8th Information Hiding Workshop*], *Springer LNCS* **4437**, 249–264 (2007).

[4] Pevný, T. and Fridrich, J., "Merging Markov and DCT features for multi-class JPEG steganalysis," in [*Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*], *Proc. SPIE* **6505**, 03–14 (2007).



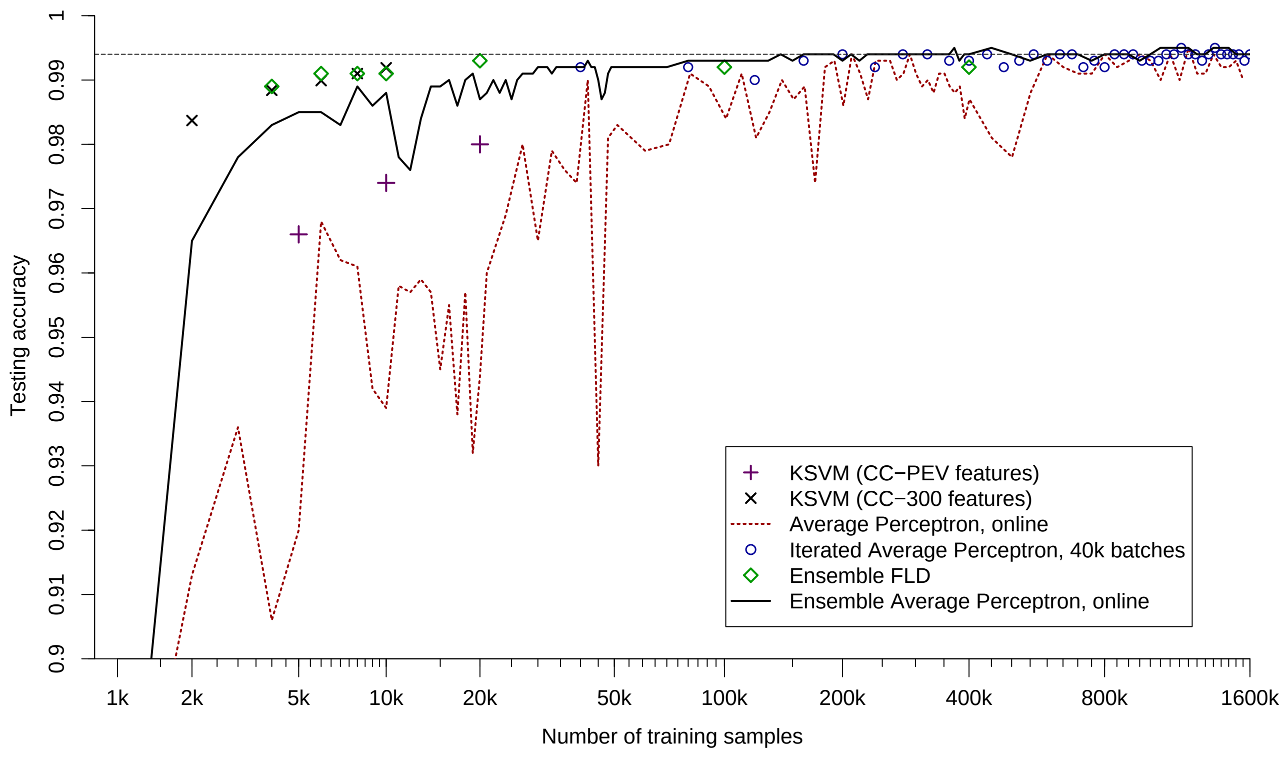


Figure 1. Combined results of all experiments. Above, payload of 0.1 bpnc. Below, payload of 0.2 bpnc. Note the nonlinear scale on the x-axis.

[5] Ker, A. D. and Lubenko, I., "Feature reduction and payload location with WAM steganalysis," in [*Media Forensics and Security XI*], *Proc. SPIE* **7254**, 0A01–0A13 (2009).

[6] Kodovsky, J., Pevný, T., and Fridrich, J., "Modern steganalysis can detect YASS," in [*Media Forensics and Security XII*], *Proc. SPIE* **7541**, 0201–0211 (2010).

[7] Bas, P., Filler, T., and Pevný, T., "Break Our Steganographic System: the ins and outs of organizing BOSS," in [*Proc. 13th Information Hiding Workshop*], *Springer LNCS* **6958**, 59–70 (2011).

[8] Lubenko, I. and Ker, A. D., "Steganalysis using logistic regression," in [*Media Watermarking, Security, and Forensics III*], *Proc. SPIE* **7880**, 0K01–0K11 (2011).

[9] Davidson, J. L. and Jalan, J., "Steganalysis using partially ordered Markov models," in [*Proc. 12th Information Hiding Workshop*], *Springer LNCS* **6387**, 118–132 (2010).

[10] Pevný, T., Fridrich, J., and Ker, A., "From blind to quantitative steganalysis," in [*Media Forensics and Security XI*], *Proc. SPIE* **7254**, 0C01–0C14 (2009).

[11] Kodovsky, J. and Fridrich, J., "Steganalysis in high dimensions: fusing classifiers built on random subspaces," in [*Media Watermarking, Security, and Forensics III*], *Proc. SPIE* **7880**, 78800L (2011).

[12] Barni, M., Cancelli, G., and Esposito, A., "Forensics aided steganalysis of heterogeneous images," in [*Acoustics Speech and Signal Processing*], *Proc. IEEE ICASSP '10*, 1690 –1693 (2010).

[13] Pevný, T., Bas, P., and Fridrich, J., "Steganalysis by subtractive pixel adjacency matrix," in [*Proc. 11th ACM workshop on Multimedia and Security*], *ACM MM&Sec '09*, 75–84 (2009).

[14] Fridrich, J., Kodovský, J., Holub, V., and Goljan, M., "Steganalysis of content-adaptive steganography in spatial domain," in [*Proc. 13th Information Hiding Workshop*], *Springer LNCS* **6958**, 102–117 (2011).

[15] Pevný, T., *Kernel Methods in Steganalysis*, PhD thesis, Binghamton University, SUNY (2008).

[16] Fridrich, J., [*Steganography in Digital Media: Principles, Algorithms, and Applications*], Cambridge University Press (2009).

[17] Langford, J., Smola, A., and Zinkevich, M., "Slow learners are fast," *Journal of Machine Learning Research* **1**(2099), 1–9 (2009).

[18] Halevy, A. Y., Norvig, P., and Pereira, F., "The unreasonable effectiveness of data," *IEEE Intelligent Systems* **24**(2), 8–12 (2009).

[19] Tong, S. and Koller, D., "Restricted bayes optimal classifiers," in [*Proc. 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*], *AAAI/IAAI*, 658–664, AAAI Press / The MIT Press (2000).

[20] Ker, A. D., "The square root law in stegosystems with imperfect information," in [*Proc. 12th Information Hiding Workshop*], *Springer LNCS* **6387**, 145–160 (2010).

[21] Zinkevich, M., Weimer, M., Smola, A. J., and Li, L., "Parallelized stochastic gradient descent," in [*Proc. 24th Neural Information Processing Systems*], *Advances in Neural Information Processing Systems*, 2595–2603 (2010).

[22] Fridrich, J., Pevný, T., and Kodovský, J., "Statistically undetectable JPEG steganography: dead ends challenges, and opportunities," in [*Proc. 9th ACM workshop on Multimedia and Security*], *ACM MM&Sec '07*, 3–14 (2007).

[23] Chang, C.-C. and Lin, C.-J., *LIBSVM: a library for support vector machines* (2001). `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] Freund, Y. and Schapire, R. E., "Large margin classification using the perceptron algorithm," *Mach. Learn.* **37**, 277–296 (December 1999).